Security Model for SOA in Banking

Nabeel Muhammed Kottayil

Abstract— **The rapid advancement of technology has changed the way the world operates. Absorption in Service Oriented Architecture (SOA) is rapidly expanding in the banking world due to the many advantages it offers such as interoperability re-usability, flexibility, reliability, efficiency and manageability.**

Banks require proper integration and collaboration among organizational systems to provide correct and efficient services. On the contrary, they include heterogeneous and distinct infrastructures, applications and systems. One of the important features of SOA is the many users of different subsystems and their communication and cooperation in activities. In this architecture, resources and services are often provided in sharing for different users. But, such features in service oriented architecture have brought about some challenges to the technology, one of which is 'security'. Criticality of the banking domain combined with the complexity that commonly exists is such huge working environment push the security problem in SOA projects to the edge. Therefore i am proposing Service Clark-Wilson Integrity Model (SCWIM), a top down integrity model for SOA capable of describing sufficient conditions to protect data integrity in any SOA implementation based on the original Clark-Wilson Integrity Model. My model can form the basis for system security audits and assist SOA architects in developing banking systems that protect data integrity as well as providing guidance for evaluating existing SOA systems.

Index Terms – **Banking, Security, Service Oriented Architecture, Service Clark-Wilson Integrity Model.**

I. INTRODUCTION

Globalization continues to pressure organizations for increased collaboration within their value chains. The banking industry, a virtual backbone for all other industries, feels this pressure both within their industry and with those they serve. Collaboration demands technology integration, and approaches so far have resulted in redundancy and inefficiency wired together with inefficient systems. Through a modular approach to underlying technology integration, service-oriented architecture (SOA) can help minimize redundancy, inflexibility and inefficiency in crucial banking processes such as payments, multichannel integration, account opening and electronic fund transfer among banks. In the banks, the systems require full integration for providing efficient and accurate services considering the interoperability and security. The service-oriented architecture has been considered an appropriate paradigm to create integration and collaboration between information systems and its goal is to estimate the customers' needs. It should be considered that banking includes various and dissimilar subsystems, infrastructures and technologies, which affect the security sector. This has constituted a set of security requirements and standards for banking, in which the security requirements of each layer are reviewed considering the organizational service-oriented architecture model in banking needs and discussed in accordance with the pertinent standard requirements.

The popular protocol and data format in use for web services are SOAP and XML .In SOA implementations a registry system called as UDDI is used for web service discovery and publication. Web servers advertise services using a well defined interface or WSDL file and uses SOAP messages as a communication mechanism between web services. The creation of a business processes from composite web services and the coordination between these services in the SOA environment follows one of two strategies; either an orchestration strategy or a choreography strategy. In the choreography strategy interactions between web services are specified from a global perspective, whereas in the orchestration strategy it is specified from a single point of view of one participant, the orchestrator. We are mainly interested in securing SOA networks that are implemented using web service technology since it is the most common implementation technology in use today. In this chapter we will try to give a brief summary of SOA and its definition, requirements, security, security models, and attacks. In this paper, the security issue in service-oriented architecture and providing a proposed security model have been discussed.

II. SERVICE ORIENTED ARCHITECTURE

The reality in IT enterprises is that infrastructure is heterogeneous across operating systems, applications, system software, and application infrastructure. Some existing applications are used to run current business processes, so starting from scratch to build new infrastructure isn't an option. Enterprises should quickly respond to business changes with agility; leverage existing investments in applications and application infrastructure to address newer business requirements; support new channels of interactions with customers, partners, and suppliers; and feature an architecture that supports organic business. SOA with its loosely coupled nature allows enterprises to plug in new services or upgrade existing services in a granular fashion to address the new business requirements, provides the option to make the services consumable across different channels, and exposes the existing enterprise and legacy applications as services, thereby safeguarding existing IT infrastructure investments.

SOA is based on principles that support a flexible approach for the realization of distributed systems that interacts across

Manuscript received September 25, 2015

Nabeel Muhammed Kottayil, System Development Assistant Manager, Information Technology Department, Eskan Bank, Seef , Kingdom of Bahrain

domain boundaries, be they inside the company or scattered among a multitude of business partners co-operating in order to accomplish a common business goal.

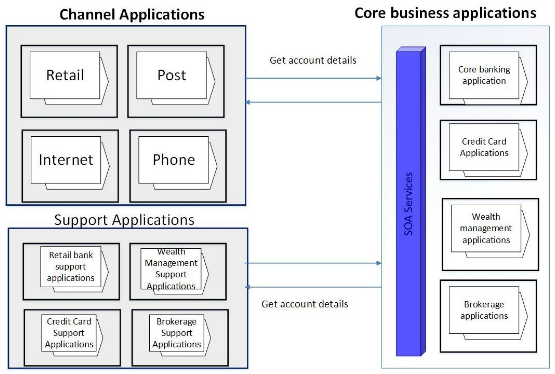
Channel Applications
Core business applications
Retail
Post
Internet
Phone
Get account details
Core banking application
Credit Card Applications
Wealth management applications
Brokerage applications
SOA Services
Support Applications
Retail bank support applications
Wealth Management Support Applications
Credit Card Support Applications
Brokerage Support Applications
Get account details

Figure 1.1 SOA Services supporting banking applications

The concept of SOA is based on the notion of services. Services packages application functionality and make it available through interfaces. Because a component functionality is specified through its interface, thereby hide implementation detail, it said to be loosely coupled with other systems that may access its services. This abstraction separates service description from the execution environment

III. SECURITY IN SAO

Security in the SOA environment involves securing all elements of the SOA network services, messages, data stores. Due to the decentralized and distributed nature of SOA networks and the use of web services, different services are distributed across different platforms and enterprises which means that data flows in all directions and needs to be protected at all times [10]. Many users can use the available services concurrently leading to global consistency problems if not carefully managed. In such an open environment, distinguishing legitimate service requests from illegitimate ones, securing the integrity of the messages in transit, the data and meta data stored, and the communication channel becomes a challenge. The traditional mechanisms that are available are typically based on point-to-point security, meaning that they will protect a communication between one endpoint and another endpoint. And they will enable authentication between those two endpoints but they will not provide the kind of security required to protect information when it is sitting in an intermediary or propagate that authentication information to the next stop in the process. In a point to point communication the technologies for providing integrity are well known (e.g. SSL, TLS) but for a distributed environment like SOA where the message might go through multiple intermediaries we need to think about end to end communication security instead.

IV. SERVICE CLARK-WILSON INTEGRITY MODEL (SCWIM)

CWIM's structure is similar to SOA's structure and it captures most of its requirements and characteristics. However, despite the similarities between CWIM and SOA, CWIM cannot be directly applicable to SOA. Therefore, we propose Service Clark-Wilson Integrity Model (SCWIM) competent of combine the notion of a service as an integration of sub-services, service contract, consistency and concurrency, transaction sequencing and service dependencies into the original certification and enforcement rules. In this section we will suggest some modifications and extensions to these rules in an attempt to achieve the objective of preserving data integrity. As the previous definitions of SOA imply, SOA networks consist of a collection of services collaborating to perform a business processes. For each initiated request there is a base service that will be in charge of contacting other services whom in turn might contact more services, this process can continue until the request is fulfilled. We will refer to this base service as the Root Service (RS).
All data items in the network are classified into CDIs (constrained data items) or UDIs (unconstrained data items) based on how their integrity affects the overall integrity of all services and data items. Each service is responsible for a set of CDIs. And all updates and changes performed on these CDIs are done by this service. Two different types of procedures are used to check and maintain the integrity of all services and CDIs, these are IVPs (Integrity Verification Procedures) and TPs (Transformation Procedures). The IVPs ensure that all CDIs in the network meet the integrity constraints of the system before the start of any transaction (i.e. the system is in a valid state). The TPs implement the functionality of the SOA and are required to move all services and CDIs from one state to another. Each root service is considered to be a Well Formed Service(WFS) if it leaves all sub-services CDIs in a valid state after the completion of a TP and ensures the global consistency of all CDIs in all services to still be valid despite failure of/or unexpected input to any service or sub-service.
Each TP consists of a set of Service Transformation Procedures (STPs). These STPs represent sub transactions between sub-services as follows. STP1 is formed of all transactions taking place between all sub-services from the root service to the final service. STP2 is the set of transactions from the second called service to the end and so on. In order for the TP to move all services from one valid state to another valid state, it is necessary but not sufficient that all STPs be completed successfully and all services be left in a valid state. If all sub services are mutually in a valid state before and after a well formed service, then the system should be in a valid state after fulfilling the requested service. Figure 2.1 is a sample SOA network that shows the relations between all of the previously defined concepts and definitions. Each service contains a set of CDIs like service S9, but to reduce the clutter.
In the diagram this was not shown in other services. The figure also show STPs are laid out inside a TP. Before the start of the TP, the IVPs validate the integrity of all services and CDIs in the diagram to make sure that the global consistency is maintained and all services and CDIs are in a valid state.
We realize that in general an IVP is at best challenging to implement by some types of applications. Based on the previous definitions the following set of modified rules form the base for the Service Clark-Wilson Integrity Model (SCWIM) we are proposing for SOA. These rules are classified into two different types: enforcement rules and certification rules. The enforcement rules are enforced by all applications and services that use the model, whereas the certification rules are certified by the security officer or system owner with respect to an integrity policy.

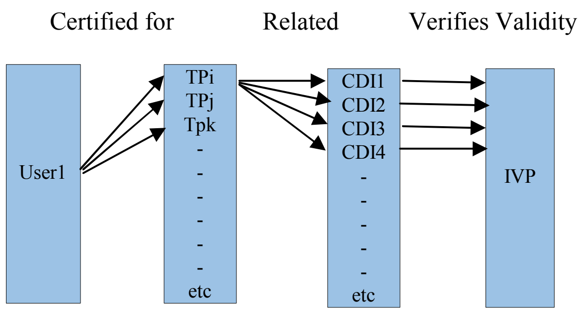

Figure 1.2 The relationship between different entities in Clark-Wilson Integrity Model

A. Encapsulation

Encapsulation is the packing of data and functions into a single component. Encapsulation increases the decoupling between different services and as a result increases flexibility. The following rule captures the encapsulation requirement in SOA.

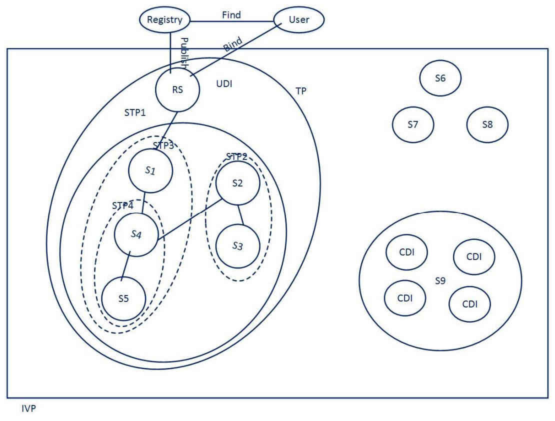

Figure 1.2 SCWIM entities interactions

CertificationRule1: All CDIs are associated with a service and each service is responsible for performing updates and changes to its CDIs.

B. Service Contract

Since SOA networks might be widely distributed among different networks, infrastructures and geographic locations with a diverse mix of new and old technologies, they need a well defined service

Contract that will maintain consistency and integrity of the data and manage the communication between different services regardless of the combination of systems or technologies involved.

CertificationRule2: Each service must be certified to have a well defined service contract that captures all of these certification and enforcement rules. It must be certified to maintain the consistency and integrity of the data regardless of the combinations of networks, infrastructures, locations and technologies involved.

C. Concurrency and Consistency Control

In SOA environment multiple transactions can take place at the same time between different services that are not necessarily dependent on each other. Although this concurrency can improve the performance of independent services, it can cause problems in the case of dependent services. This can lead to consistency problems if not carefully managed by a set of concurrency and consistency rules specified for this purpose. In the original Clark-Wilson model, concurrency was not a problem because we were dealing with one host and one copy of each CDI. When one CDI is involved in a transaction, no other transaction is able to use it until the first transaction is completed and the state of the CDI is updated. Concurrency would not be a problem, unless we are dealing with different copies of the same CDI on different machines or services, because this might affect the global consistency where some of the services might not be using the last updated version of that CDI. It is not possible to enforce consistency constraints after each action. One may need to temporarily violate the consistency of the system state while modifying it. What is important here is to maintain the global consistency of all services and CDIs once the TP is completed especially in the case of concurrency.

CertificationRule3: If a CDI can be used and updated by two different services S1 and S2 simultaneously. Then both services must be certified to ensure the mutual consistency of the updated CDI and all other CDIs.

CertificationRule4: Concurrent TPs must be certified to maintain the global consistency of all services and CDIs once they are completed.

D. Authentication and Authorization

Users and services might have several identities to use in different networks or for different services. These identities must be authenticated and subject to the same security controls, as described in the following rule.

Enforcement Rule1: Each service must authenticate the identity of all subjects attempting to execute a TP whether these subjects were users or services. As well as authenticating all propagations of these identities across all dependent services for this TP. The notion of an STP needs to be incorporated in all of the relationships used in the authorization process to make sure that each STP is certified and authorized to use certain CDIs. The subject in these relations can be identified by a userID or a serviceID and the data items can be local to the service or a reply from another service. The following two rules capture that. *CertificationRule5:* All TPs and STPs must be certified to be valid. That is, they must take a CDI to a valid final state, given that it is in a valid state to begin with. For each TP, and each set of CDIs that it may manipulate, the security officer, must specify a ”relation”, which defines that execution. A relation is thus of the form: (TPi, STPi, (CDIa, CDIb, CDIc)), where the list of CDIs defines a particular set of arguments for which the TP has been certified.

Enforcement Rule2: Each service must maintain a list of relations of the form: (SubjectID, TPi, STPi, (CDIa, CDIb, CDIc)), which relates a subject, a TP, an STP and the data objects that these TPs and STPs may reference on behalf of that user. It must ensure that only executions described in one of the relations are performed. In addition, it is necessary to ensure that all manipulations on data items are not done arbitrarily but in a constrained manner that will maintain the integrity of this data item and other data items to guarantee the global consistency of all services and sub services. The concept of well-formed transactions captures that in the following rule.

CertificationRule6: All STPs must be certified to be part of a well formed TP. This certification rule should capture all the dependencies between services and sub services.

E. Separation of Duty

The principle of separation of duty implies that the agent responsible for creating or certifying a well formed transaction must not be allowed in the process of implementation or execution of that transaction. *CertificationRule7:* The list of relations maintained by each service must be certified to meet the separation of duty requirements.

EnforcementRule3: Only the subjects permitted to certify entities may change the list of such entities associated with other entities: specifically, those subjects associated with a TP. An agent that can certify an entity may not have any execute rights with respect to that entity.

F. Transaction Sequencing

To ensure the integrity of the business process, transactions must be performed in a specific sequence. In many cases, applications implement a first in/ first out (FIFO) queue by waiting for each transaction to be completed before the next one in the queue is processed (processing transactions serially). For example, a billing application cannot compute the total cost of a bill before it has looked up the rates that apply to the customer and computed subtotals for each different category of services. However, in SOA the sequence of transactions relies on the dependencies between services. For example by looking at figure 1 we can see that service S4 can't be executed before services S2 and S3 are executed. This means that the order in which STPs are executed must be as follows: STP3, STP4, STP2, STP1. The following rule captures this and guarantees the global consistency of all services and data items. CertificationRule8: For each TP, the order in which STPs are performed must be certified to maintain the global consistency of all the services and data items.

G. Service Dependencies

Service dependencies can take place between any number of services in order to fulfill a single request, and as a result increases the number of data items being manipulated raising the probability of putting the system in an invalid state due to failure in one or more of the sub-services. To make the process of ordering, auditing and recovery possible each service must maintain a dependencies table that records all dependencies between different services in a service network as shown in the following rule. It is also possible to have one service be responsible for maintaining this dependencies table. EnforcementRule4: Each service must maintain a dependencies table recording all dependencies between different services in a service network in the abstraction of: (Service ID, Depends on (Sa ID, Sb ID, Sc ID)).

H. Auditing

Many CDIs are involved in the fulfillment of a request. Validating these CDIs after each step is not a convenient process nor does it guarantee that the overall system is in a valid state. Therefore, if a sub-service failed to respond to a service call due to any reason, there should be a recovery mechanism to roll back all manipulations done before the failure in order to return the system to the previous valid state it was in. *CertificationRule9:* All TPs must be certified to write to an append-only CDI (the log) all information necessary to permit the nature of the operation to be reconstructed.

I. Integrity Verification and System State

Whenever all the CDIs meet the integrity constraints of the system, the system is said to be in a valid state. The IVPs are responsible for checking that all CDIs in the SOA network are is in a valid state before the beginning of any new transaction. CertificationRule10: All IVPs must properly ensure that all CDIs are in a valid state at the time the IVP is run. In the case of performing a business process in a SOA environment the IVP of the root service is valid if the IVPs of all sub-services are valid.

CertificationRule11: A well-formed service (WFS) must be certified to ensure that all sub-service CDIs remain in a valid state and that the global consistency of CDIs is valid despite failure of any service or sub-service.

J. Filtering

The original CW model requires that all inputs whether from users or responses from other services be filtered at the interface before being used. The filtering process filters the data items into CDIs or UDIs based on how they affect the integrity of the system. All inputs entered by users to the services are considered UDIs and needed to be upgraded to CDIs or otherwise rejected.

CertificationRule12: Any TP or STP that takes a UDI as an input value must be certified to perform only valid transformations, or else no transformations, for any possible value of the UDI. The transformation should take the input from a UDI to a CDI, or the UDI is rejected. If a TP started with a valid state and all certification and enforcement rules were applied, then it is guaranteed that none of the services will enter a bad state due to any reason.

CONCLUSION

Security models may be implemented in several ways to satisfy the integrity requirements specified in a security policy. Model implementations describe how specific mechanisms can be employed in a banking system to ensure that the goals of the security policy are met. The Service Clark-Wilson model emphasizes how integrity is key to the banking environment and it seeks to develop better security systems for that environment. Service Clark-Wilson Integrity Model (SCWIM is to protect data integrity in any SOA implementation based on the original Clark-Wilson Integrity Model. This model can form the basis for system security audits and assist SOA architects in developing systems that protect data integrity as well as providing guidance for evaluating existing SOA systems in banking environment.

REFERENCES

[1] Yuan E. and Tong J. Web services security: What's required to secure a service- oriented architecture An Oracle White Paper, October 2006.

[2] Chris Peltz. Web services orchestration and choreography. Computer, 36(10):46–52, October 2003.

[3] Service oriented architecture – Revolutionizing today's banking systems, IBM Institute for Business value white paper by Jay DiMare and Richard s.MaDavidWalend.

[4] Understanding service oriented architecture. DeveloperNetwork,November2006.
[5] Ramarao Kanneganti and Prasad A Chodavarapu. SOA Security. Manning Publications, January 2008.
[6] DavidSprottandLawrenceWilkes. Understanding service oriented architecture. TheArchitecture Journal, January 2004.
[7] Yuan E. and Tong J. Attributed based access control (abac) for web services. Proceedings of the 2005 IEEE International Conference on Web Services (ICWS'05), 00:561–569, 2005.
[8] Information Security Management Handbook, Fourth Edition, Volume 3 By Harold F. TiptonM. Young

Author Nabeel Muhammed Kottayil was born in Vettathur, Kerala,India. After completing his graduation in Physics from Calicut University, he completed his MCA from Periyar University. He did his MBA(Information System) from Annamalai University. He has more than 10 years of system development experience in banking domain. He is currently Assistant Manager – System Development at Eskan Bank, Bahrain. He is member of MCP. He is also Microsoft Certified Solution Developer (MCSD).